

The PC's in room 025 Hayes Hall have Matlab installed. To run it, click Start, Programs, Applications, Matlab. The **command window** appears, with  $\gg$  as the prompt. The commands below should be typed at the  $\gg$  prompt. Don't type the line numbers or the comments at the ends of the lines.

### Matlab basics

1: 2*347	Matlab can be used as a calculator.
2: 2^999	
3: sin(1)	
4: format long	Show more than 4 decimal places.
5: sin(1)	
6: format	Go back to showing just 4 decimal places.
7: x=sin(1)	You can store values with variable names.
8: x	This displays the value of x
9: x = [6 7 8 9 10]	x is a <b>1 by 5 vector</b> ; several numbers stored under the same name.
10: x(4)	Show the 4th element of x.
11: x(2:4)	The 2nd, 3rd, and 4th elements of x.
12: y = [1 4 9 16 25];	The semicolon suppresses the output of the command.
13: plot(x,y);	Plot (x(1),y(1)), (x(2),y(2)), etc. and connect with lines.
14: hold on	The next plot command will be added to the previous one.
15: plot(x,y,'r*');	Plot red stars at each data point.
16: clf	Clear the figure so we can make a new plot.
17: x = 0:10	An easier way to type x=[0 1 2 3 4 5 6 7 8 9 10]
18: y = x.^2	Square every element of x.
19: plot(x,y,'k')	Plot y against x in black.
20: help plot	Basic information about the plot command. For more documentation, open <a href="http://www.mathworks.com/access/helpdesk/help/helpdesk.html">http://www.mathworks.com/access/helpdesk/help/helpdesk.html</a>

### Random numbers and statistical tools

21: x=rand(10,1)	x is a collection of 10 random numbers, uniformly distributed between 0 and 1. x is like a column of cells in a spreadsheet. It is called a <b>vector</b> or a 10 by 1 <b>matrix</b> .
22: x+5	Add 5 to each entry of x.
23: 100*x	Multiply each entry of x by 100.
24: round(100*x)/100	Round each value of x to two decimal places.
25: x=rand(10,7)	A 10 by 7 matrix of random numbers. To repeat this, or any other command, hit the up arrow and then press enter.
26: x=rand(1000,1)	Go ahead, try something larger than 1000.
27: x=rand(1000,1);	Use a semicolon to suppress the output.
28: max(x)	
29: mean(x)	
30: sum(x)	
31: median(x)	
32: cumsum(x)	A vector of cumulative sums of entries of x.
33: y=sort(rand(10,1))	Sort another 10 random numbers.
34: plot(x)	Plot the values of x against numbers 1, 2, ...
35: hist(x)	Make a histogram of the values of x.
36: hist(x,30)	Use 30 bins instead of the default 10.
37: y=-log(x);	Take the natural logarithm of the entries of x.
38: hist(y,30)	Most of the values of y are near 0, but some are as large as 6. The numbers in y have what is called the <b>exponential distribution</b> .
39: z=randn(1000,1);	Generate 1000 normally distributed numbers

40: `hist(z,30)` The histogram is bell-shaped. You might want to try this with more than 1000 numbers.

### Writing a Matlab program to simulate asexual reproduction

You are going to create a new Matlab program called `Asexual_reproduction.m`, which will conduct a simulation of asexual reproduction similar to what we did in class.

At the top of the command window is a blank white sheet; click on that to open a new editor window. In that window, type the commands below on separate lines, then save the program as `Asexual_reproduction.m` by clicking on the disk icon.

```
41: GP = 'ABCDEFGHJKLMNOPQRST';           The initial state of the gene pool
42: for g = 1:1000,       Execute the following block of commands 1000 times. The first time, g
    will equal 1, then 2, then 3, etc.
43:     for i = 1:length(GP),           Loop through each character of GP
44:         if rand < 1/3,   rand is a random number between 0 and 1, so this will happen 1/3
            of the time
45:             GP(i) = ' ';           Replace character i with a blank space; the individual dies
46:         end                       End the if statement
47:     end                             End the loop over entries of GP
48:     fprintf('Generation %d deaths : %s\n', g, GP);   Print out the state of the gene
    pool after removing individuals that die
49:     for i = 2:length(GP),           Loop through characters to fill in surviving genotypes
50:         if GP(i) == ' ',           If the individual died,
51:             GP(i) = GP(i-1);       Use the genotype to the left
52:         end
53:     end
54:     if GP(1) == ' ',               If the left-most individual died,
55:         GP(1) = GP(end);           Replace it with the far-right genotype
56:     end
57:     for i = 2:length(GP),   Same as above, in case several individuals on the left side died
58:         if GP(i) == ' ',
59:             GP(i) = GP(i-1);
60:         end
61:     end
62:     fprintf('Generation %d gene pool: %s\n', g, GP);
63:     pause                           Wait for the user to press a key
64: end                                 End of each generation
```

To run the program, type `Asexual_reproduction` at the command prompt. If it can't find the program, follow the instructions below to set the path to include the folder where you saved the program.

### Binomial distribution

```
65: y = binopdf(0:20,20,2/3);           Binomial probabilities
66: y(1)                                 Probability that 0 live
67: (1/3)^20                             Probability that 0 live, as we computed it
68: bar(0:20,y);                          Make a bar graph of these probabilities
69: title('Binomial probabilities with 20 trials and probability 2/3 of success')
```

### Using user-defined Matlab programs

70: Download `MatlabPrograms.zip` from the course website and extract all the files into a folder on your disk or other storage device. Now point Matlab to that folder by pulling down File, Set Path, Add Folder ..., and navigate to the folder where you put the files. Now you can run programs from that folder by simply typing their name at the prompt in the command window, as you will see below. Don't double click the files, however. That may launch another program called Mathematica, which also uses a `.m` extension on its filenames.

## Coin flipping with Matlab

- 71: `3<5` The answer is 1 because the inequality is satisfied.
- 72: `5<3` The answer is 0.
- 73: `x=rand(20,1)` A vector of 20 random numbers between 0 and 1.
- 74: `x>0.5` Check the inequality for each entry of x.
- 75: `z=1+(x>0.5)` z will equal 1 or 2. This is like flipping coins, with 1 for heads and 2 for tails.
- 76: `show(z,'HT')` Use a program called `show`, which needs two arguments, a vector of integers greater than 0 and a string to tell what letters to use to display each number.
- 77: `show(1+(rand(1500,1)>0.5),'HT')` Generate 1500 coin flips. If the output runs off the right side of the screen, you can pull down File, Preferences, Command Window, and check the box labeled Wrap lines. Now press the up arrow and return, repeatedly. That's a lot of coin flips! What is the longest run of heads that you can find?
- 78: `show(1+(rand(1500,1)>0.5),'0-')` You can make that question easier to answer with different characters. Use the up arrow so you can simply edit the previous command.
- 79: `show(1+(rand(1500,1)>0.7),'0-')` Using a cutoff of 0.7 will make the probability of getting a 0 0.7 and the probability of getting a - 0.3.
- 80: `show(ceil(6*rand(1500,1)),'123456')` A similar technique can be used to simulate rolls of a die.

## Maximum likelihood

- 81: `BinomialLikelihood` Binomial trials are situations in which each trial can result in either success or failure. For instance, when flipping a coin, heads could be called success and tails could be called failure. It is common to have binomial trials but not to know what the probability of success is. The best way to find out is to do many trials and keep track of the number of successes, then estimate the success probability. This program will prompt you for the number of trials and the number of successes, then it will graph the likelihood function. You will be able to enter additional data from additional trials and see how the likelihood function changes. As more trials are completed, the likelihood function narrows, meaning that fewer success probabilities are consistent with the data. The likelihood function also gets shorter. Why?

## Bayesian analysis of binomial trials

- 82: `BayesTwoCoins` There are two coins in a box. One is fair, the other lands heads only 40% of the time. This program chooses a coin at random from the box and flips it to try to determine which coin it is.
- 83: `BayesManyCoins` There are many, many coins in a box, only one of them fair. All the others land heads different percentages of the time, from 0 to 100. This program chooses a coin at random from the box and flips it to try to determine the fraction of time the coin will land heads up.

## Plotting functions and finding the maximum

- 84: `clf`
- 85: `ezplot('(x^6)*(1-x)^4', 0, 1);` Easy way to plot a function from 0 to 1.
- 86: `[x,fval] = fmaxbnd('(x^6)*(1-x)^4',0,1)` Easy way to find the maximum value of a function and where it occurs. The function `fmaxbnd` is a user-defined program that calls the Matlab program `fminbnd`, which finds the minimum value of a function over a bounded interval. For some reason, Matlab only makes it easy to find the minimum of a function, but if you take the negative of the function you want, the maximum becomes the minimum.
- 87: `hold on`
- 88: `plot(x,fval,'r*');` Add the maximum point to the plot.
- 89: `grid on` Add horizontal and vertical lines to the plot.
- 90: `title('Likelihood function for 6 heads and 4 tails');`
- 91: `xlabel('Probability of getting heads on one flip');`
- 92: `ylabel('Likelihood');`